RFC 7828 : The edns-tcp-keepalive EDNS0 Option

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 avril 2016. Dernière mise à jour le 12 avril 2016

Date de publication du RFC : Avril 2016

https://www.bortzmeyer.org/7828.html

Historiquement, le DNS utilisait surtout UDP et le transport sur TCP ne servait qu'à des cas particuliers, comme lorsqu'un client DNS ré-essayait avec TCP de récupérer des données trop grosses pour UDP. Depuis le RFC 7766 [1], ce n'est clairement plus le cas : TCP est désormais aussi utilisable qu'UDP pour le DNS. Mais les mises en œuvre typique du DNS sur TCP ont des restrictions, par exemple des délais maximum d'attente avant de « raccrocher » qui sont trop courtes, de l'ordre de quelques secondes lorsqu'il n'y a plus de trafic sur la connexion. Ce nouveau RFC propose un mécanisme EDNS permettant au client DNS de signaler au serveur qu'il souhaite garder la connexion ouverte, si le serveur est d'accord, s'il vous plait. (Notez qu'il existe une autre façon d'indiquer le temps pendant lequel on souhaite garder la session, celle des DSO du RFC 8490 ».)

Utiliser une connexion TCP par requête DNS serait en effet très inefficace, notamment question latence <https://www.bortzmeyer.org/latence.html>. Il faudrait effectuer la triple poignée de mains TCP à chaque requête. Le DNS sur TCP n'est envisageable que si les connexions restent ouvertes et que la même connexion peut servir pour plusieurs requêtes. Dans ce cas, lorsque la connexion a déjà été ouverte, la latence peut être aussi faible qu'avec UDP, voire meilleure (dans le cas de données de grande taille, par exemple avec DNSSEC, plus besoin d'essayer d'abord sur UDP). Reste une question pour le serveur DNS : faut-il laisser la connexion ouverte dès qu'il n'y a plus de requêtes à traiter ? Cette nouvelle option EDNS `edns-tcp-keepalive` permet au client DNS d'indiquer que, lui, il est prêt à réutiliser la connexion et qu'il souhaite qu'on la garde ouverte.

Le serveur souhaite économiser ses ressources (mémoire, notamment). Contrairement à UDP, TCP nécessite que le serveur garde un état. Imaginons que chaque connexion TCP prenne une entrée dans un grand tableau des connexions TCP en cours. Si c'était « *"open bar"* » pour les clients, qu'ils puissent ouvrir autant de connexions qu'ils veulent, et les garder éternellement ensuite, le tableau serait vite

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7766.txt

rempli, menant à un déni de service. (Et si vous pensez qu'à la place du tableau, une structure de données dynamique, grossissant automatiquement, résoudrait le problème, rappelez-vous que la mémoire du serveur est finie...) Le serveur choisit donc s'il garde les connexions ouvertes, et combien de temps.

Au passage, la section 1 du RFC rappelle aussi qu'UDP a des problèmes sérieux, notamment parce qu'il permet des attaques par réflexion <https://www.bortzmeyer.org/attaques-reflexion.html> et parce qu'il implique, pour les grosses données, de la fragmentation, qui est en général une source d'ennuis (voir par exemple « "*Fragmentation Considered Poisonous*" <http://arxiv.org/abs/1205.4011> de Herzberg et Shulman). D'où cet encouragement, depuis des années, à utiliser de plus en plus TCP <https://www.bortzmeyer.org/dns-over-tcp.html>.

La section 3 est la partie centrale du RFC, la spécification de la nouvelle option. Elle utilise EDNS (RFC 6891). Elle permet d'indiquer le délai d'attente avant la fermeture de la connexion, délai qui commence à courir lorsque la connexion devient inactive (« "*idle*" », cf. la section 3 du RFC 7766).

L'option se nomme `edns-tcp-keepalive` et a reçu le code 11 (stocké dans le registre des options EDNS <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-parameter
Elle prend un seul argument, encodé sur deux octets, `TIMEOUT` qui indique la durée du délai de garde, exprimée en unités de 100 millisecondes. Cette valeur n'apparait que dans les réponses du serveur, pas dans les requêtes du client. Si `TIMEOUT` vaut 600, cela signifie que les connexions restent ouvertes pendant une minute après la dernière requête reçue.

Ensuite, une fois cette option définie, comment on s'en sert ? Elle ne doit évidemment pas être utilisée sur UDP, où elle n'aurait aucune signification. Le client doit utiliser cette option pour signaler qu'il souhaiterait garder cette connexion ouverte (il ne doit pas indiquer de valeur pour `TIMEOUT`). Le serveur utilise cette option pour indiquer le temps pendant lequel il va garder la connexion (rappelez-vous que le client peut demander ce qu'il veut, c'est le serveur qui décide). Le client est censé « raccrocher » juste avant l'expiration de ce délai. Une valeur de 0 pour `TIMEOUT` signifie « raccroche tout de suite » (un serveur l'envoie, par exemple, lorsqu'il est soumis à un fort stress et n'a plus de ressources disponibles.)

Au passage, si le serveur n'a plus envie de supporter cette connexion, pourquoi ne raccroche-t-il pas lui-même, au lieu de demander au client de le faire ? C'est parce que le pair TCP qui ferme la connexion le premier passe ensuite en état `TIME-WAIT` (RFC 793, section 3.5) et que cela l'oblige à garder un état pendant deux fois la durée de vie maximale d'un segment TCP (soit quatre minutes en tout). Il vaut donc mieux demander au client de le faire. Par contre, après avoir envoyé le `TIMEOUT=0`, le serveur va avoir un délicat arbitrage à faire entre attendre que le client raccroche et fermer de lui-même la connexion (et donc garder une prise en état `TIME-WAIT`) quand le client n[Caractère Unicode non montré [2]]obtempère pas suffisamment vite.

La section 5 du RFC recommande d'ailleurs aux serveurs de suivre l'activité de leurs clients et de « punir », d'une façon ou d'une autre, ceux qui violeraient ces règles, par exemple en continuant à envoyer des requêtes alors qu'on leur a demandé de raccrocher.

J'insiste mais cela vaut la peine d'être dit et répété : cette option `edns-tcp-keepalive` ne diminue pas la liberté des clients et des serveurs DNS. En cas de problème (tableau des connexions TCP en cours presque plein, par exemple), clients et serveurs sont libres de couper les connexions TCP plus tôt, ce qu'on appelle le « "*TCP session management*" ».

Comme d'habitude avec toute nouvelle option, on peut s'attendre à des problèmes avec les stupides "*middleboxes*". Une requête contenant cette option peut ainsi être jetée alors qu'elle ne l'aurait pas été sans l'option. Une stratégie de repli comme celle de la section 6.2.2 du RFC 6891 peut donc être utile.

Merci à Kim-Minh Kaplan pour d'utiles précisions sur TCP.

2. Car trop difficile à faire afficher par LaTeX